PRINCIPI DI SISTEMI OPERATIVI

ESERCIZIO del 5 LUGLIO 2004

Una società di trasporto autovetture si avvale di un **camion cicogna** per trasportare le auto tra il magazzino e la concessionaria. Il camion può contenere al massimo **MAX auto**. L'**autista** del camion parte da una località solo quando il camion è pieno, cioè sono state caricate **MAX** auto; per poter essere scaricate, le auto devono attendere che l'autista arresti il camion nella concessionaria.

Si implementi una soluzione usando il costrutto monitor per modellare il **camion cicogna** e i processi per modellare le **auto** e l'**autista** e si descriva la sincronizzazione tra i processi. Nella soluzione si massimizzi l'utilizzo delle risorse. Si discuta se la soluzione proposta può presentare starvation e in caso positivo per quali processi, e si propongano modifiche e/o aggiunte per evitare starvation.

```
program camion_cicogna

const     MAX = ...; { capacità della cicogna }
type      dir = (P, A, viaggio);  { stato della cicogna }
          { P -> partenza, A -> arrivo }

type auto = process
begin
          c.carica;
          <preparati ad uscire>
          c.scarica;
end

type autista = process
begin
      repeat
        c.parti;
        <fai il viaggio >
        c.arresta;
        < torna indietro >
      until false
end
```

```
type cicogna = monitor

{ variabili del monitor }
var  stato : dir;
     { stato della cicogna }
     n_auto : integer;
     { numero di auto in cicogna }
     coda : array[loc] of condition;
     { code su cui sospendere in attesa della cicogna }
     coda_viaggio : condition;
     { auto in cicogna in attesa di arrivare }
     coda_cicogna : condition;
     { cicogna in attesa di partire }

procedure entry carica
begin
     if stato <> P then { se la cicogna non è presente }
          coda[l].wait;
     n_auto ++ ; { sono entrato }
     { se abbiamo raggiunto il numero minimo }
     if (n_auto = MAX) then
          { faccio partire la cicogna }
          coda_cicogna.signal;
     { mi sospendo in attesa di arrivare }
     coda_viaggio.wait;
end

procedure entry scarica
begin
     n_auto -- ; { siamo arrivati }
end
```

```
procedure entry parti
begin
      { imposto la direzione }
      stato := P;
      { sveglio le auto in attesa di entrare }
      for i := 1 to MAX do
            coda[l].signal;
      { se non c'è il numero minimo di auto }
      if (n_auto < MAX) then
            coda_cicogna.wait;
      stato := viaggio; { ripartiamo! }
end

procedure entry arresta
begin
      { imposto la direzione }
      stato := A;
      { sveglio tutti gli auto in attesa di uscire }
      for i := 1 to n_auto do
        coda_viaggio.signal;
end

begin { inizializzazione delle variabili }
      n_auto := 0;
      stato := P;
end

var  c: cicogna; { il nostro monitor }
      a1, a2, ... : auto;
      a: autista;

begin end.
```

Starvation

La soluzione proposta non presenta starvation